On the DMA Mapping Problem in Direct Device Assignment

Ben-Ami Yassour **Muli Ben-Yehuda** Orit Wasserman

benami@il.ibm.com muli@il.ibm.com oritw@il.ibm.com

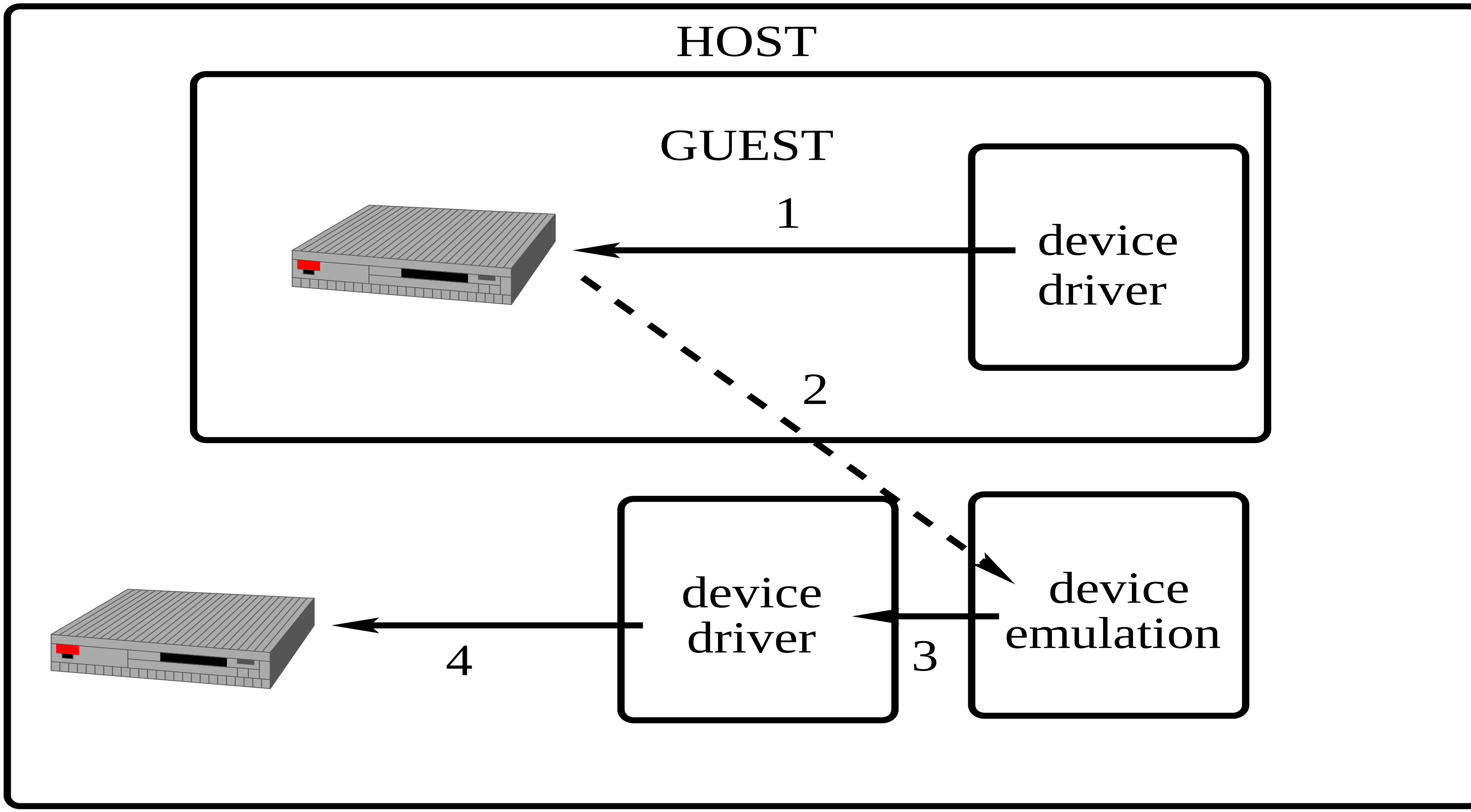
IBM Research – Haifa

Happy Birth Day Tomer Yassour

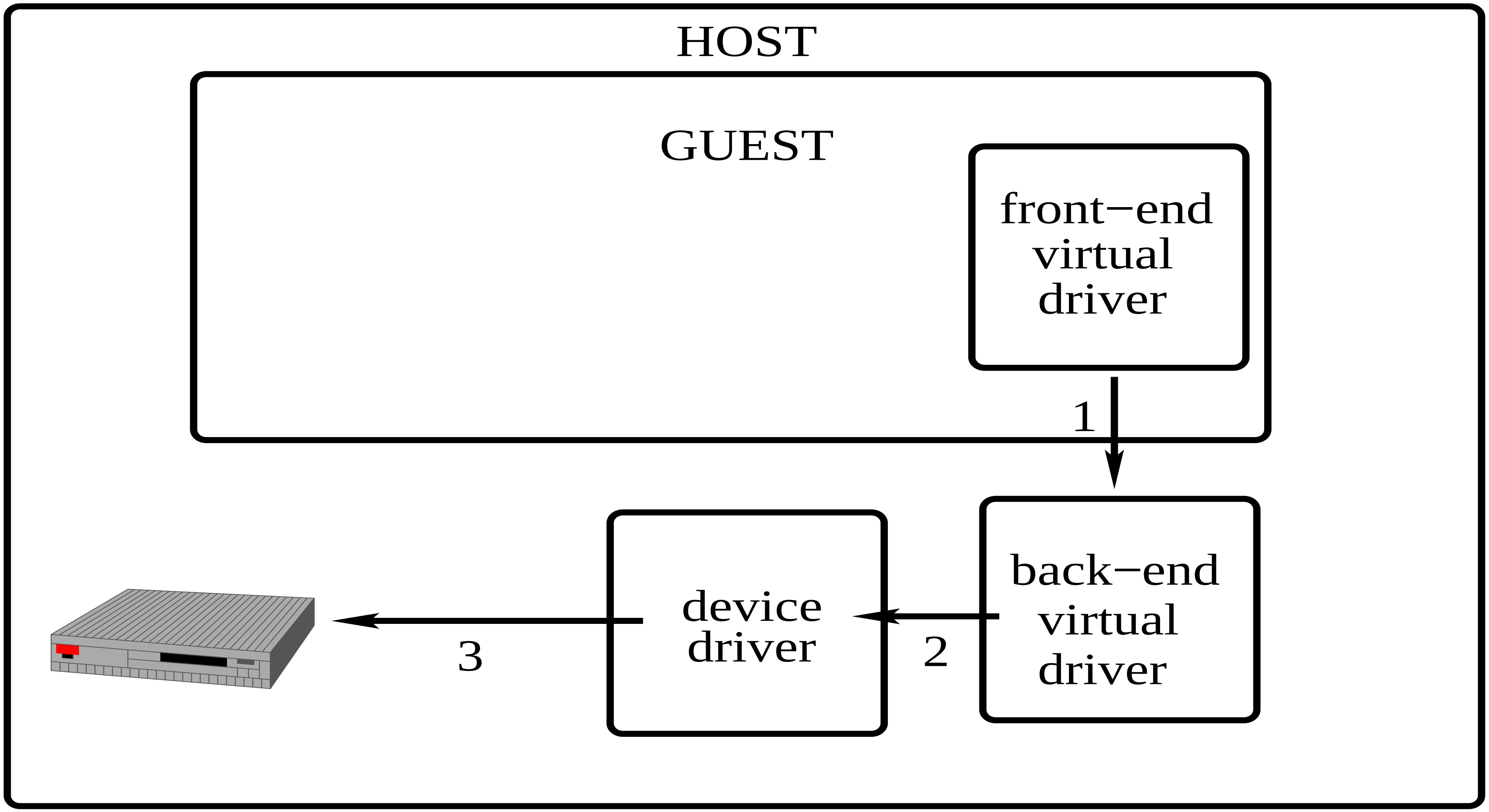
Virtual Machine I/O

- I/O virtualization is the final frontier
- How many of you would run your production database in a VM today?
- Virtual machines use one of three models for I/O:
  - Device emulation
  - Para-virtualized drivers
  - Device assignment

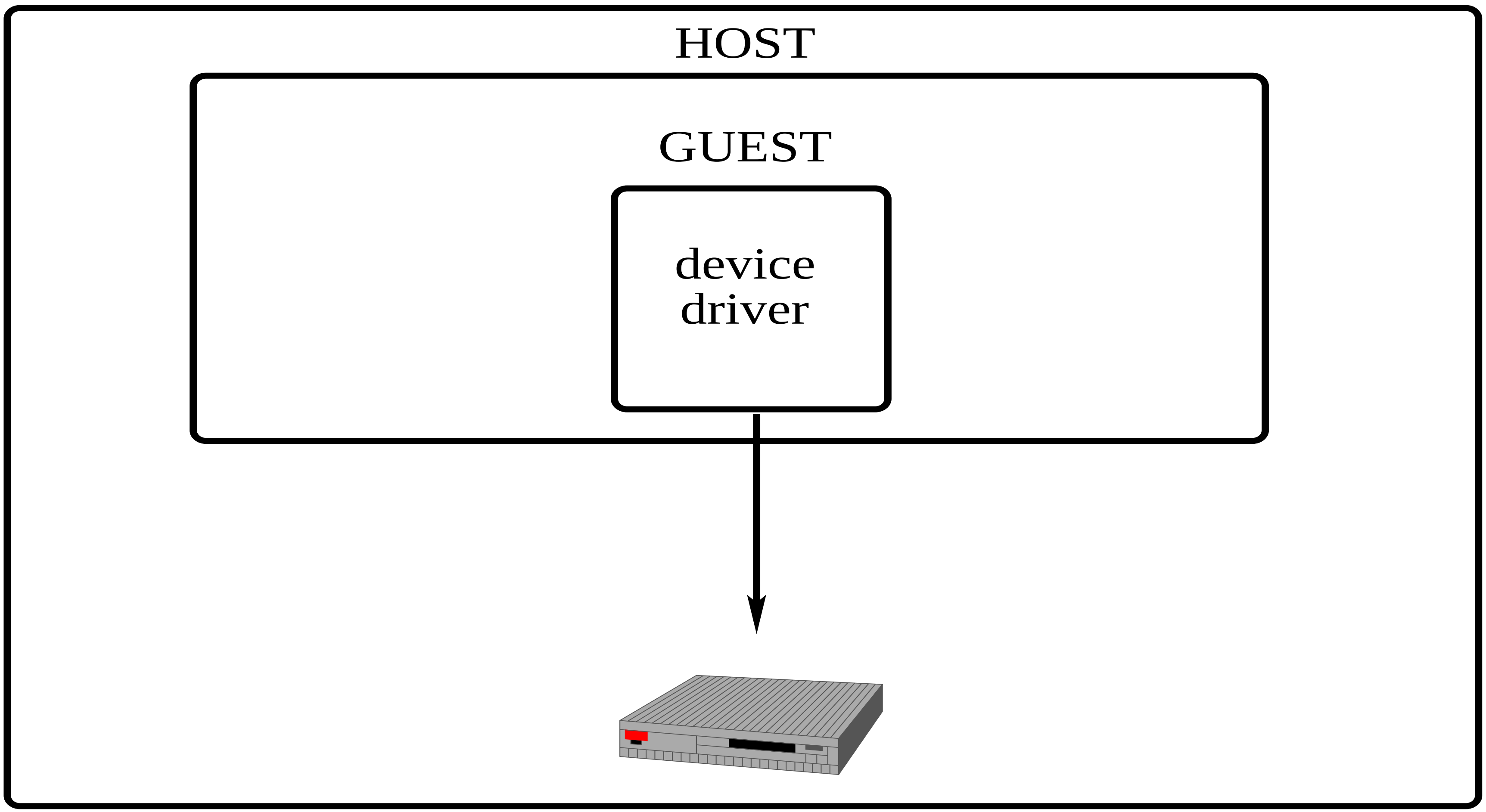
I/O: Device Emulation



I/O: Para-virtualized Drivers



I/O: Direct Device Assignment



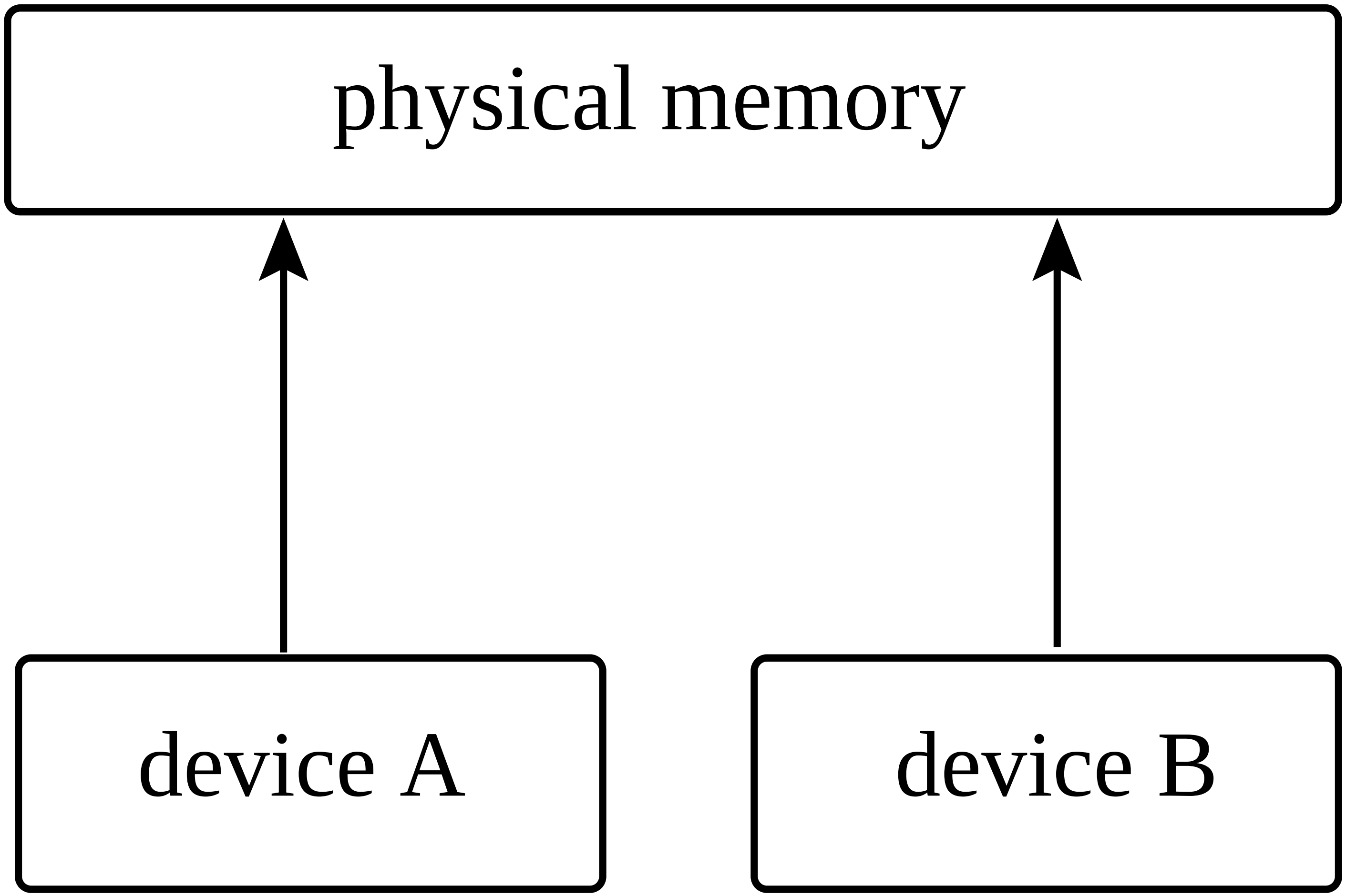
I/O: Direct Device Assignment cont'

- Give VM direct access to a hardware device
- Without any software intermediaries between the virtual machine and the device
- Examples:
  - Legacy adapters [Ben-Yehuda06]
  - Self-virtualizing (SRIOV) adapters [Liu06], [Willman07]
- $\rightarrow$ Best performance[Santos08,09]—but at a price.

The Linux/KVM Hypervisor

- A hypervisor extension for the Linux kernel [Kivity07]
- Makes extensive use of Intel and AMD hardware virtualization extensions
- Full featured, open source, and hacker friendly
- `http://www.linux-kvm.org`

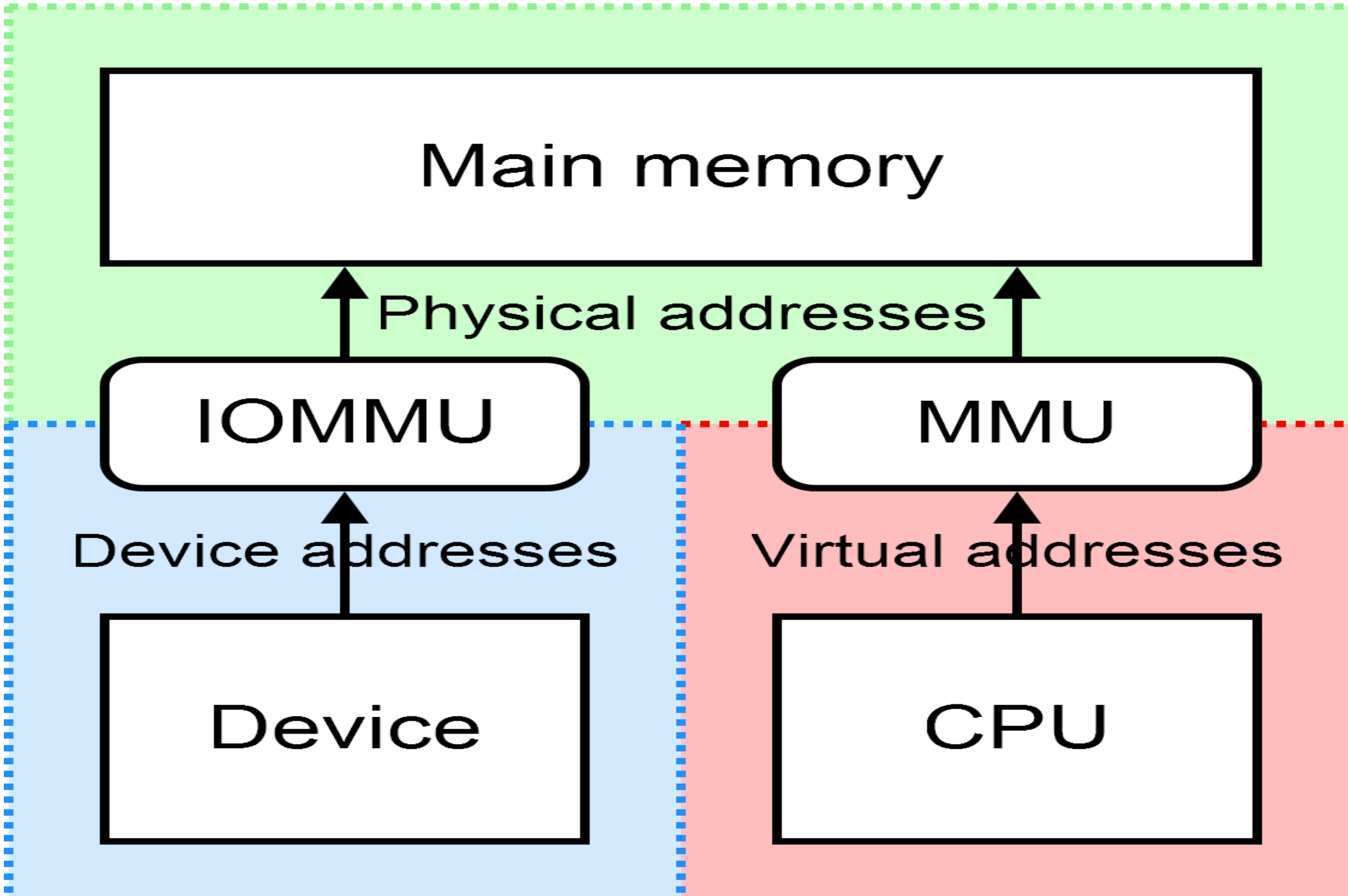
DMA



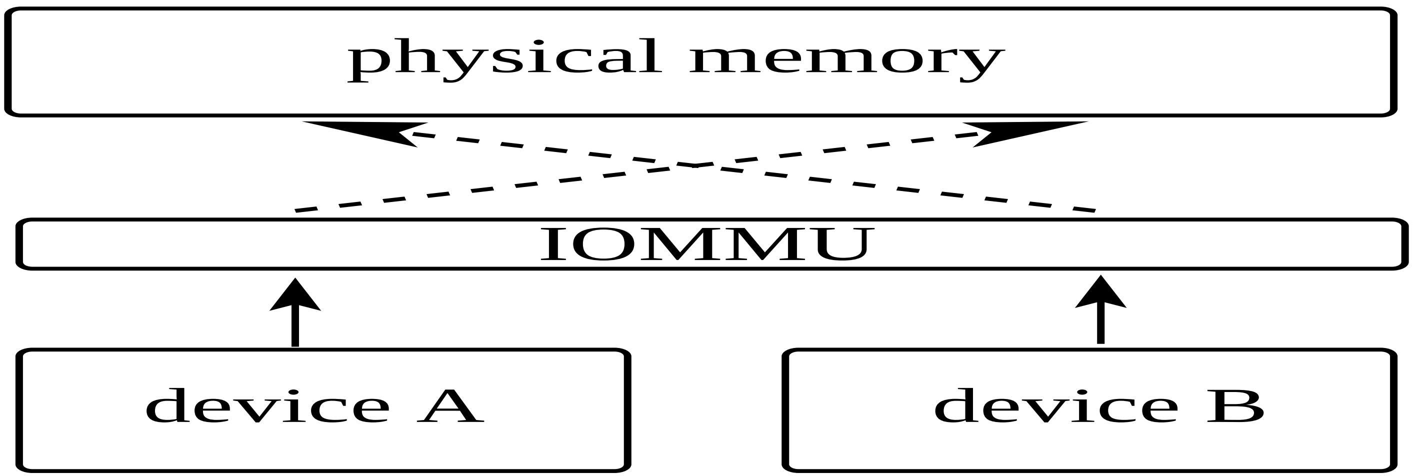
DMA Security

- Untrusted guest programs a device, without any supervision.
- Device is DMA capable (all modern devices are).
  - Which means the guest can program the device to overwrite any memory location.
- ... including where the hypervisor lives ... game over.

IOMMU

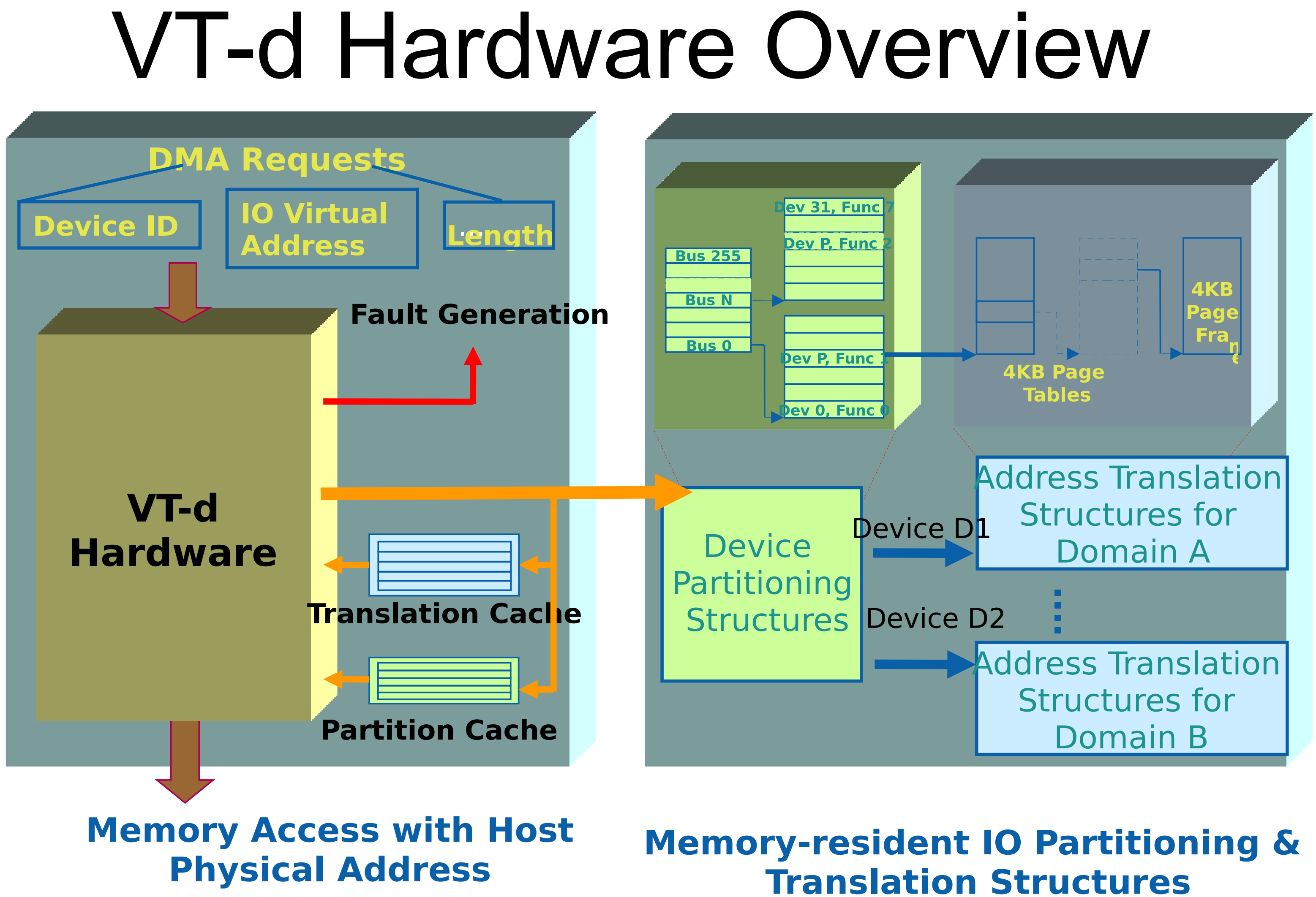

Main memory
Physical addresses
IOMMU
MMU
Device addresses
Virtual addresses
Device
CPU

IOMMU to the rescue



- IOMMU—think MMU for I/O devices—separate address spaces, protection from malicious devices!
- IOMMUs enable direct assignment for VMs.
- Intra-VM vs. Inter-VM protection [Willman08]
- But: IOMMUs have costs too [Ben-Yehuda07]

The Intel VT-d IOMMU

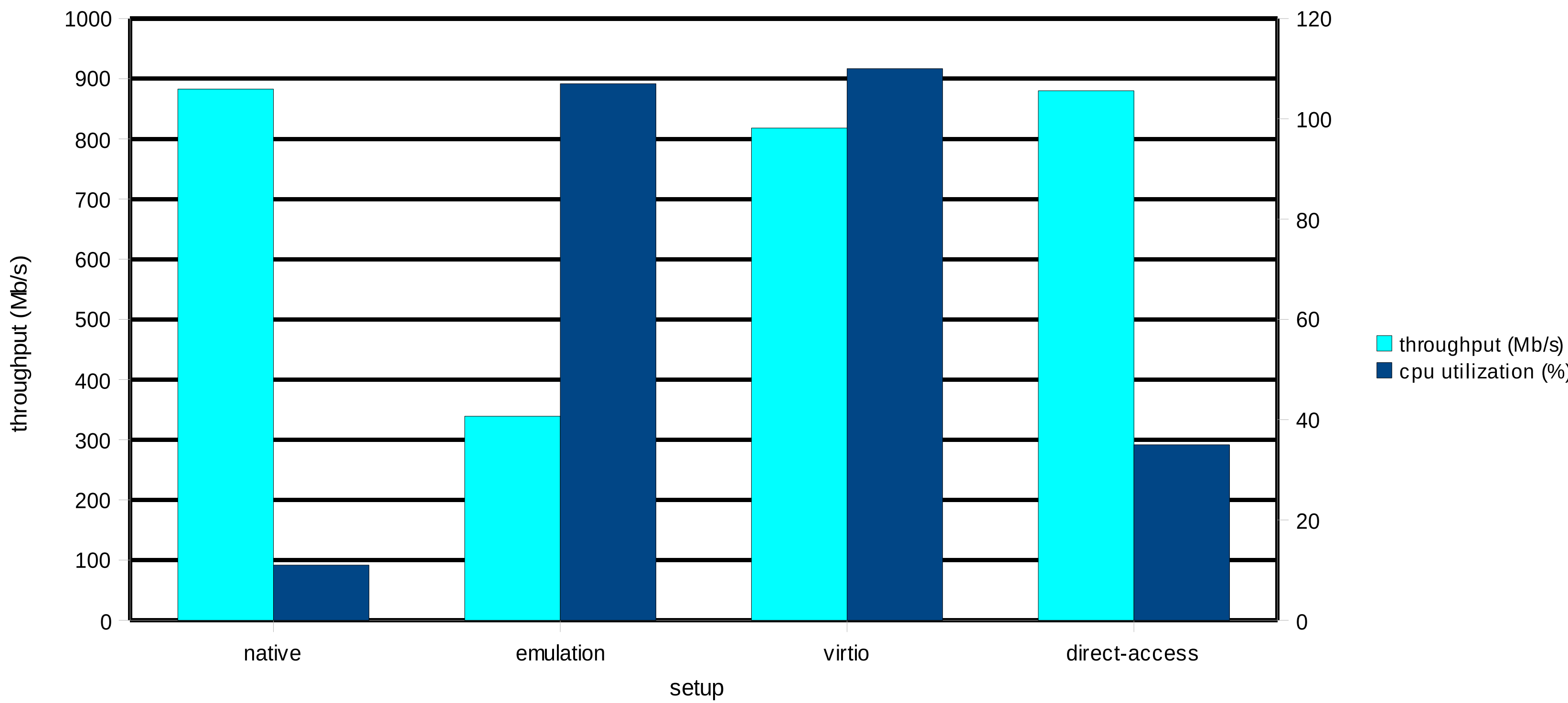


IOMMU Protection Strategies

As defined by Willman, Rixner and Cox [Willman08]:

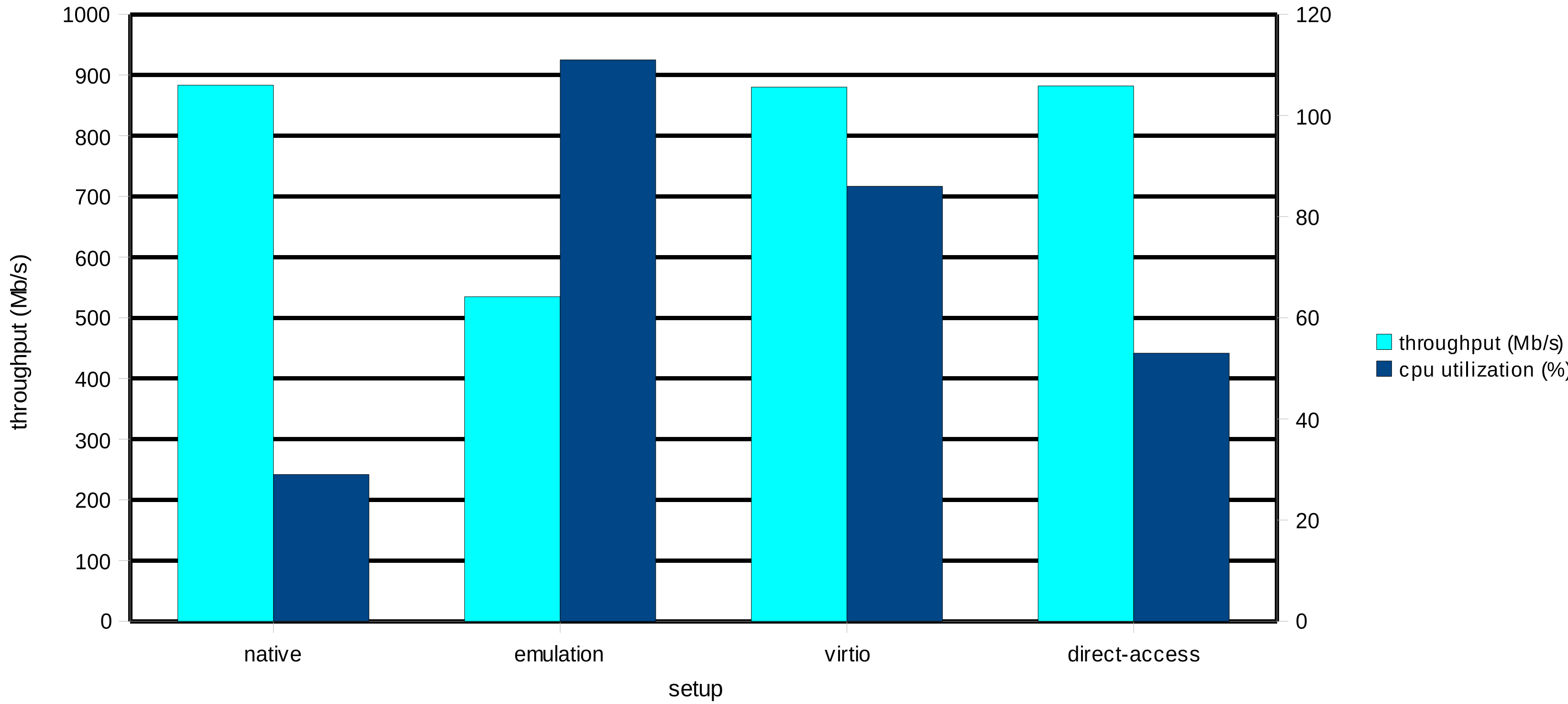
- Single-use $\rightarrow$ **Intra**-guest protection, expensive!
- Shared $\rightarrow$ Relaxed protection, expensive.
- Persistent $\rightarrow$ Inter-guest protection, pins all of memory.
- Direct-map $\rightarrow$ Inter-guest protection, no run-time cost, pins all of memory.

Direct-map Performance—Send



Note: with direct mapping, all memory is pinned!

Direct-map Performance—Receive



Note: with direct mapping, all memory is pinned!

The DMA Mapping Problem

- Single-use → expensive!
- Shared → expensive.
- Persistent → pins all of memory.
- Direct-map → pins all of memory.

The DMA mapping problem: When should a memory page be mapped or unmapped for DMA?

On-Demand Mapping Strategy

- IOMMU remappings are expensive (world switch, IOTLB flush: over 10K cycles per remapping)
- A **map-cache** for caching IOMMU mappings when they are first created. How big should it be?
- Observation: all guests have some memory pinned anyway.
- Second observation: common workloads do **not** need to use all of the guest's memory address space.
- Define a **quota** for DMA mappings: the amount of memory the guest can pin for DMA.

The Map Cache

- Mappings are created when the guest first DMAs to that page.
- Mappings are either pinned (in use) or candidates for eviction.
- Implemented using a red-black tree.
- Mappings are removed from the cache when the quota is reached and a new mapping needs to be created.

Quota control

- Cooperative guests: defining a quota that is equal to their current memory requirements leads to no run-time IOMMU remappings—best performance!
- Un-cooperative guests: smaller quota, hypervisor enforced.
- Now the question becomes: for a given quota that is smaller than the working set size, how to efficiently replace IOMMU mappings?
- Close resemblance to the classical page replacement problem.
- ... except I/O devices **do not** have page faults.

Batching Mapping Requests

- If the driver batches together multiple mapping and unmapping requests, the map cache only goes to the hypervisor once.
- Downside: requires changing the drivers.
- Piggbyacking unmap requests on top of new mappings.

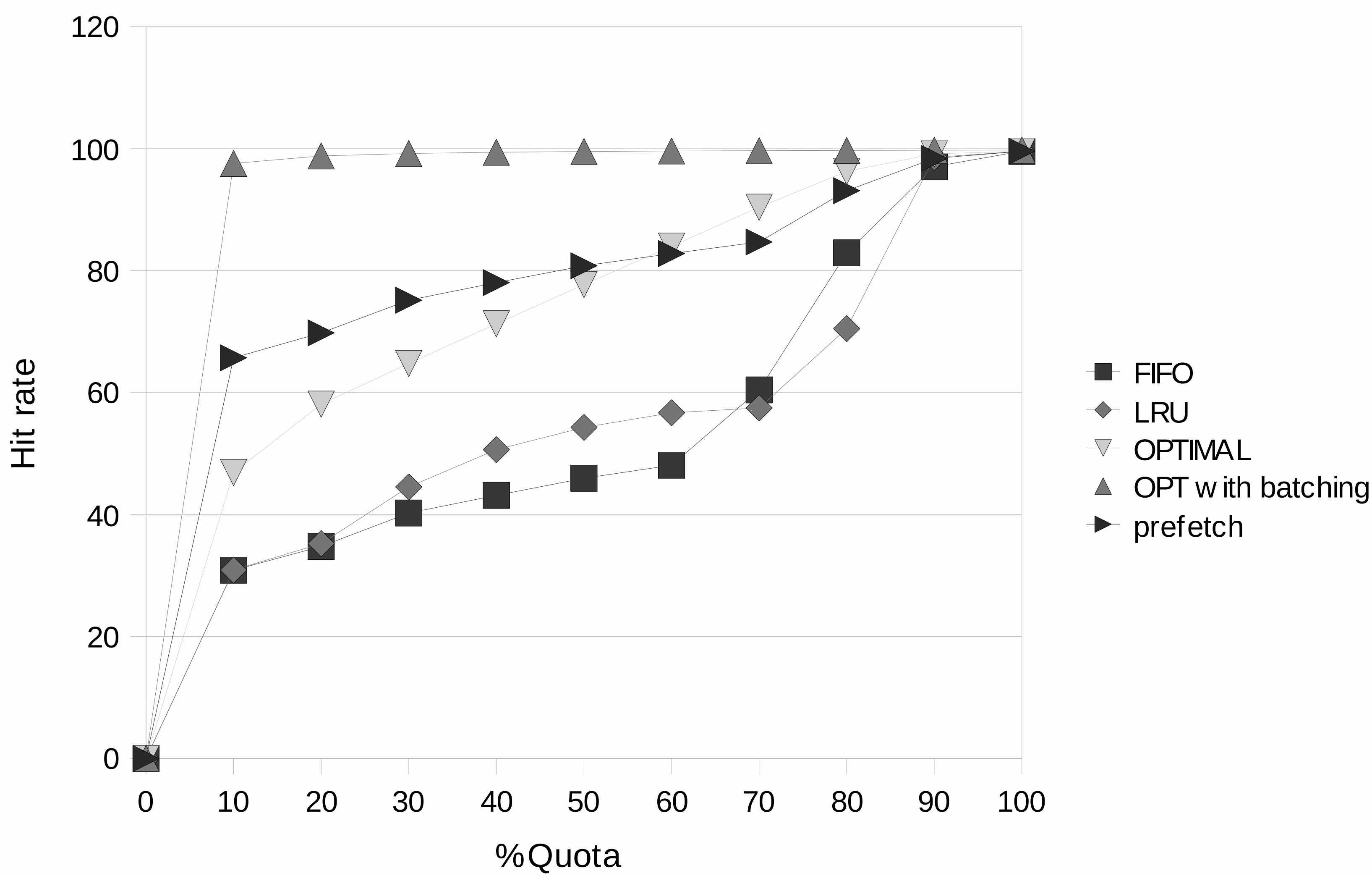
Prefetching

- No driver changes necessary!
- Same concept as FAR [Borodin91] and FARL [Fiat97] paging algorithms in the access graph model.
- Which pages were recently mapped after a given page?
- When mapping a new page, also opportunistically map its followers.
- Choose pages to evict using standard LRU.

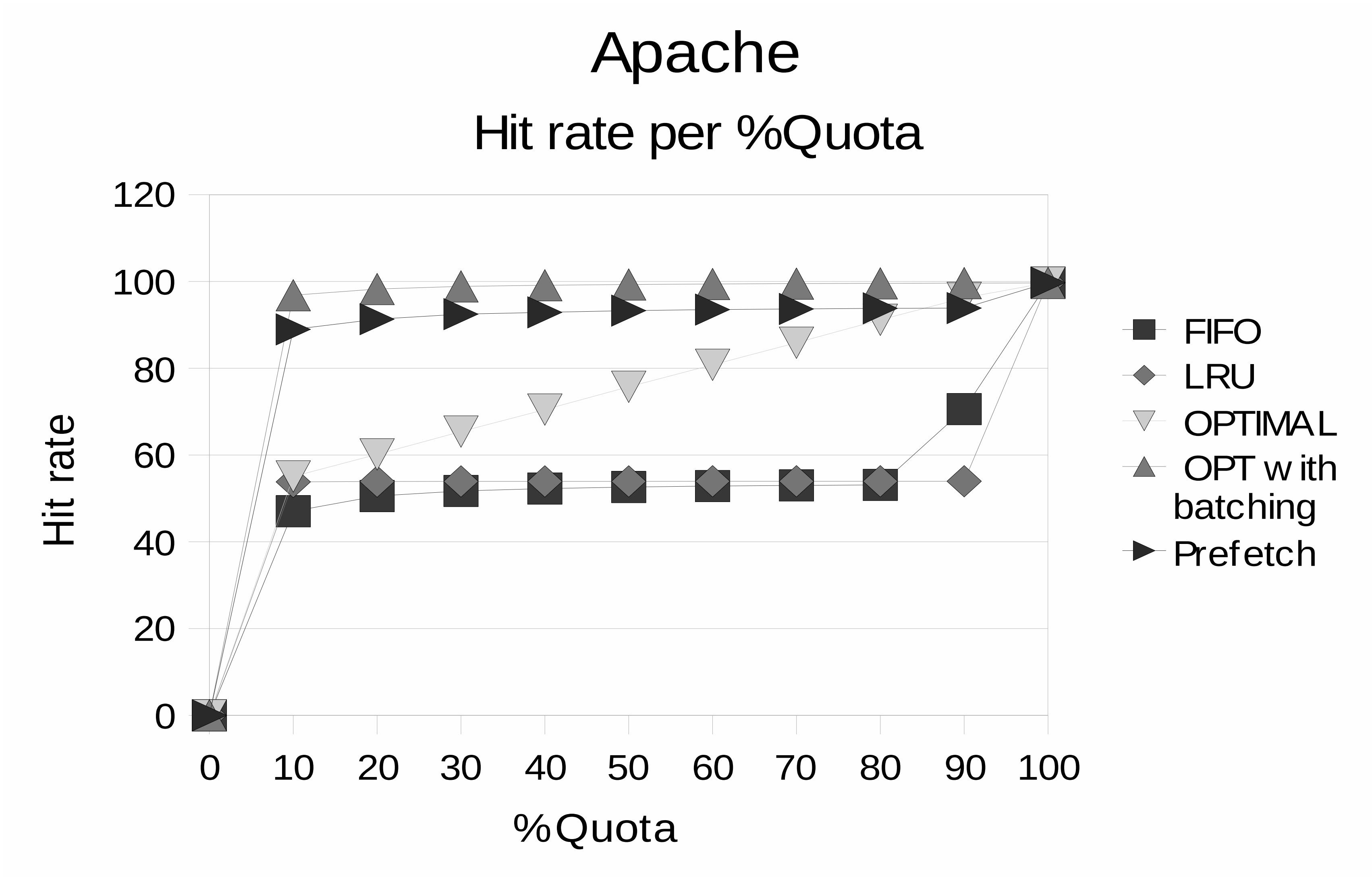
Evaluation – hit rate

- **FIFO** Evict pages in a first in first out order.
- **LRU** Evict the least recently used page.
- **OPT** Evict the page that is going to be used later then any other page in the cache. This is the optimal *offline* algorithm *without batching*.
- **Optimal batching** The optimal *offline* algorithm, but with batching.
- **Prefetching**.

netperf send hit rate per quota



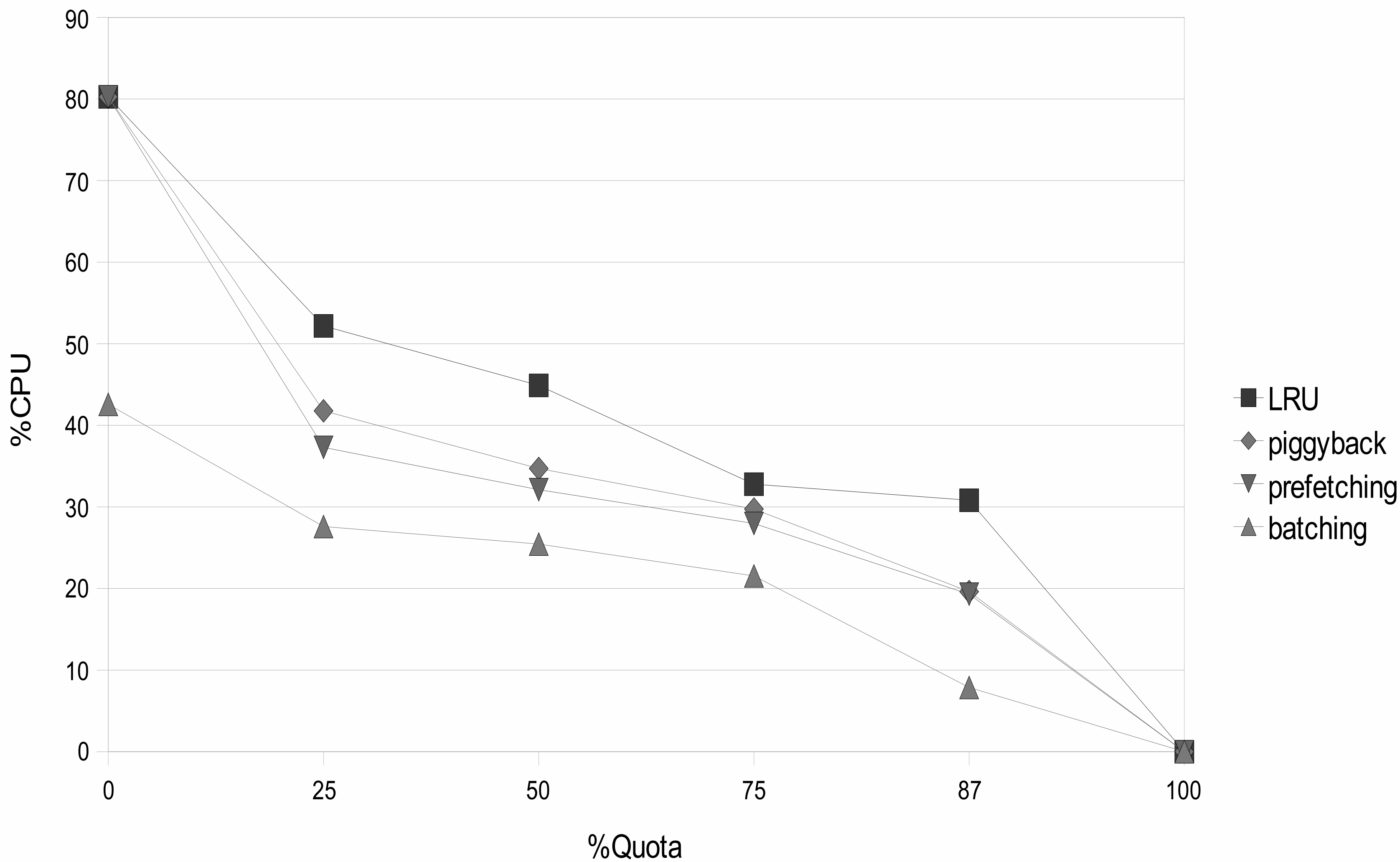
apache hit rate per quota



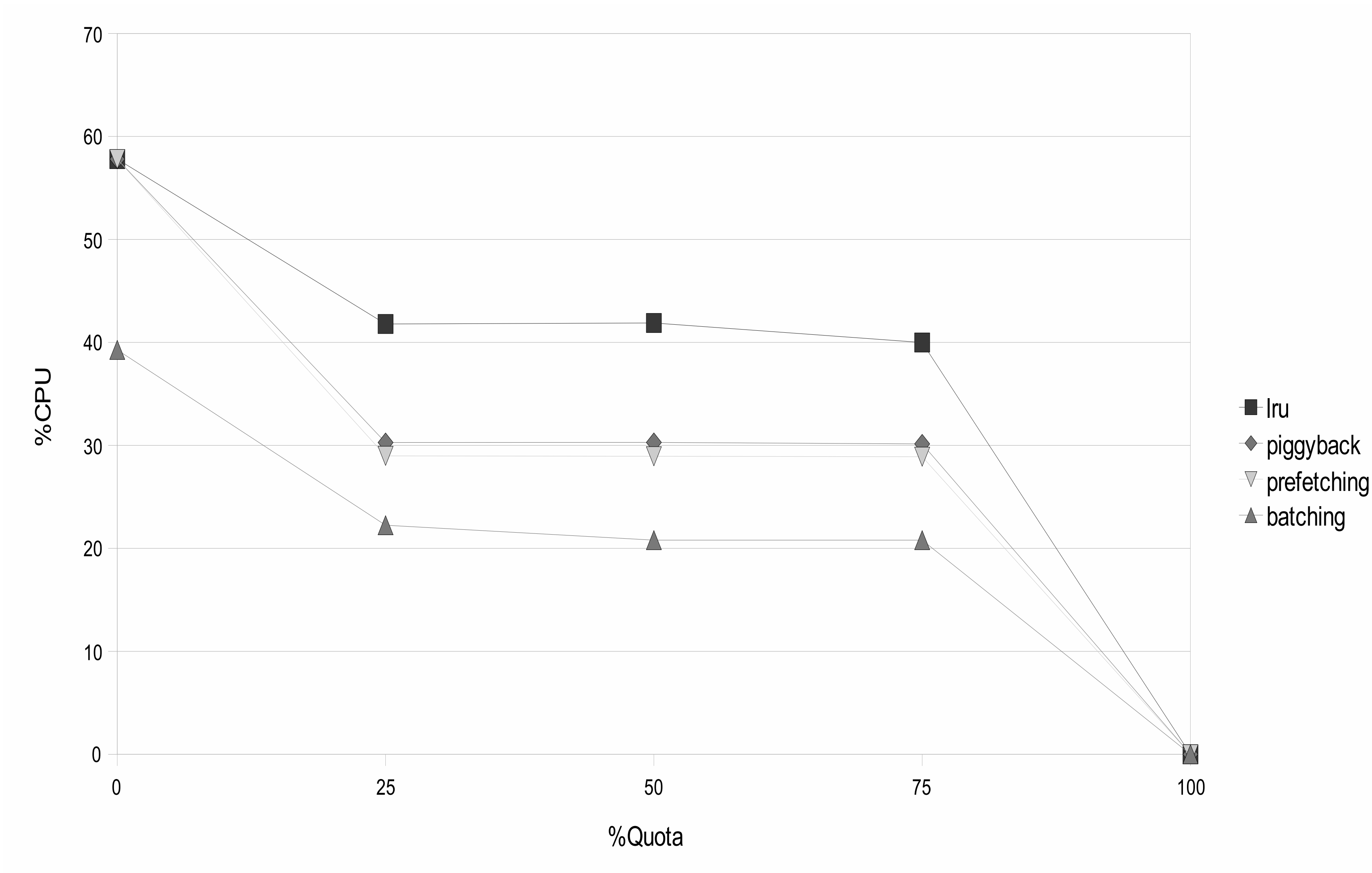
Evaluation – CPU utilization

- **LRU** Default LRU algorithm where no batching or caching is used.
- **Piggyback** Piggybacking unmaps on top of maps.
- **Prefetching**.
- **Batching** LRU with the map and unmap batching optimizations.

netperf send CPU utilization



apache CPU utilization



Summary & Conclusions

- Direct device assignment gives best performance of all I/O virtualization methods.
- ... but also poses new problems.
- In particular, how to balance DMA mapping memory consumption and performance?
- We propose the on-demand DMA mapping strategy.
  - Mappings are cached, with the size of the cache limited by a quota.
  - Remappings are batched, and new mappings prefetched.
  - Same run-time performance as persistent mapping, better memory utilization for common workloads.

Bibliography

- Barham03: “Xen and the Art of Virtualization”, SOSP ’03
- Bellard05: “QEMU, a Fast and Portable Dynamic Translator”, USENIX ’05
- Ben-Yehuda06: “Utilizing IOMMUs for Virtualization in Linux and Xen”, OLS ’06
- Ben-Yehuda07: “The Price of Safety: Evaluating IOMMU Performance”, OLS ’07
- Bhargave08: ”Accelerating two-dimensional page walks for virtualized systems”, ASPLOS ’08

Bibliography cont.

- Borodin91: “Competitive paging with locality of reference”, STOC ’91
- Chen08: “Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems”, ASPLOS ’08
- Fiat97: “Experimental studies of access graph based heuristics: beating the LRU standard?”, SODA ’97
- Liu06: “High Performance VMM-Bypass I/O in Virtual Machines”, USENIX ’06
- Kivity07: “kvm: The Kernel-Based Virtual Machine for Linux”, OLS ’07
- Popek74: “Formal Requirements for Virtualizable Third Generation Architectures”, CACM 17(7), ’74

Bibliography cont.

- Santos08: “Bridging the Gap between SW & HW Techniques for I/O Virtualization”, USENIX ’08
- Santos 09: “Achieving 10Gbps using Safe and Transparent Network Interface Virtualization”, VEE ’09
- Willman07: “Concurrent Direct Network Access for Virtual Machine Monitors”, HPCA ’07
- Willman08: “Protection Strategies for Direct Access to Virtualized I/O Devices”, USENIX ’08
- Yassour08: “Direct Device Assignment for Untrusted Fully-Virtualized Virtual Machines”, Ben-Ami Yassour, Muli Ben-Yehuda, Orit Wasserman, IBM Research Report H-0263, 2008